NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

COMPUTER IMPLEMENTATION OF ARNOTT'S FORMULATION OF THERMOACOUSTICS USING MATLAB

by

LTJG. Argirios L. Gamaletsos

December, 1993

Thesis Advisor: A.A. Atchley

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	18 November 1993.	Master's Thesis

4. TITLE AND SUBTITLE COMPUTER IMPLEMENTATION OF ARNOTT'S FORMULATION OF TERMOACOUSTICS USING MATLAB.

5. FUNDING NUMBERS

6. AUTHOR(S) Argirios L. Gamaletsos

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(maximum 200 words)*

This thesis describes a Matlab computer program that implements Arnott's formulation of thermoacoustics [W.P. Arnott , et al, "General formulation of thermoacoustics for stacks having arbitrarily shaped pore cross sections", J. Acoustic. Soc. Am. 90(6), 3228-3237 (1991)]. The program calculates the resonance frequency and quality factor of a thermoacoustic prime mover below onset of self-oscilation. The results of this analysis are compared to measured values for both a closed end and an open end prime mover and to predictions of a standing wave analysis of prime movers [A.A Atchley, "Standing wave analysis of a thermoacoustic prime mover below onset of self-oscillation, " J. Acoustic. Soc. Am. 92(5), 2907-2914 (1992)].

14. SUBJECT TERMS Prime mover; Thermoacoustics			15. NUMBER OF PAGES 72
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Approved for public release; distribution is unlimited.

COMPUTER IMPLEMENTATION OF ARNOTT'S FORMULATION OF THERMOACOUSTICS USING MATLAB

by

Argirios L. Gamaletsos
Lieutenant J.G, Hellenic Navy
B.S.E.E., Hellenic Naval Academy, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL
December 1993

Author: _____
Argirios L. Gamaletsos

Approved by: _____
Anthony A. Atchley, Thesis Advisor

_____
D.Felipe Gaitan, 2nd Reader

_____
Anthony A. Atchley, Chairman
Engineering Acoustics Academic Commitee

ABSTRACT

This thesis describes a Matlab computer program that implements Arnott's formulation of thermoacoustics [W.P. Arnott, et. al., "General formulation of thermoacoustics for stacks having arbitrarily shaped pore cross sections," J. Acoustic. Soc. Am., 90(6), 3228-3237 (1991)]. The program calculates the resonance frequency and the quality factor of a thermoacoustic prime mover below onset of self-oscilation. The results of this analysis are compared to measured values for both a closed end and an open end prime mover and to predictions of a standing wave analysis of prime movers [A.A. Atchley, "Standing wave analysis of a thermoacoustic prime mover below onset of self-oscillation," J. Acoustic. Soc. Am. , 92(5), 2907-2914 (1992)].

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First, I express my sincere "thanks" and "gratefulness" to my advisors; specially, my thanks to Prof. A.A Atchley who lead me with faith and supported me with patience in accomplishing this thesis.

Next, I thank Prof. W.P. Arnott for helping me in the construction of the Matlab program.

Finally, I specially thank the Hellenic Navy for giving me the opportunity to come to the Naval Postgraduate School and thanks to God for finishing my degree with this thesis.

I. INTRODUCTION

Thermoacoustic heat transport is a process through which an acoustic field generates, or inversely is generated from, a flow of heat. Like every engine, a thermoacoustic engine can be configured as either type of classical heat engine, a heat pump(or refrigerator) or a prime mover. The purpose of this thesis is the computer implementation of a general formulation of thermoacoustics developed by Arnott et al [Ref. 1] using Matlab. In particular, we find the theoretical values of the quality factor and the resonance frequency of a prime mover as a function of the temperature difference applied across the prime mover stack. The program determines the impedance and normalized pressure distribution along the prime mover beginning from a known value of specific acoustic impedance and normalized pressure amplitude at one end. It finds the resonance frequency and quality factor through calculation of the pressure amplitude at the opposite end as a function of frequency. The result of the computer implementation are compared with measurements made with open and closed ended prime mover [Ref. 2,3], and with the results of a standing wave analysis of prime movers [Ref. 4].

II. THEORY

The goal of this chapter is to describe the basic geometry of a prime mover and to summarize the important points of Arnott's method. Parallel with this we explain how this theory is implemented in the computer program. The reader is referred to [Ref. 1] for full details of Arnott's method.

A. ARNOTT'S METHOD.

The necessary background for this thesis can be explained with the help of Fig.1 which shows the basic construction of the thermoacoustic prime mover. It constists of a circular cross section acoustic resonator , rigidly capped at both ends. Situated within the resonator are the thermoacoustic elements consisting of an ambient heat exchanger, a prime mover stack and a hot heat exchanger. In our prime mover both the stack and heat exchangers are parallel plates spaced by a few thermal penetration depths. Complete details of the prime mover construction are given in [Ref. 2,3,4]. The traverse coordinates of the prime mover are taken to be x and y, and the longitudinal coordinate is z. The hot and cold sections are circular ducts open at the heat exchanger end and open or closed at the other end. The prime mover is divided into either 5 or 7 sections depending on the boundary conditions [Fig. 1,2]. The stack is divided into 11 subsections.

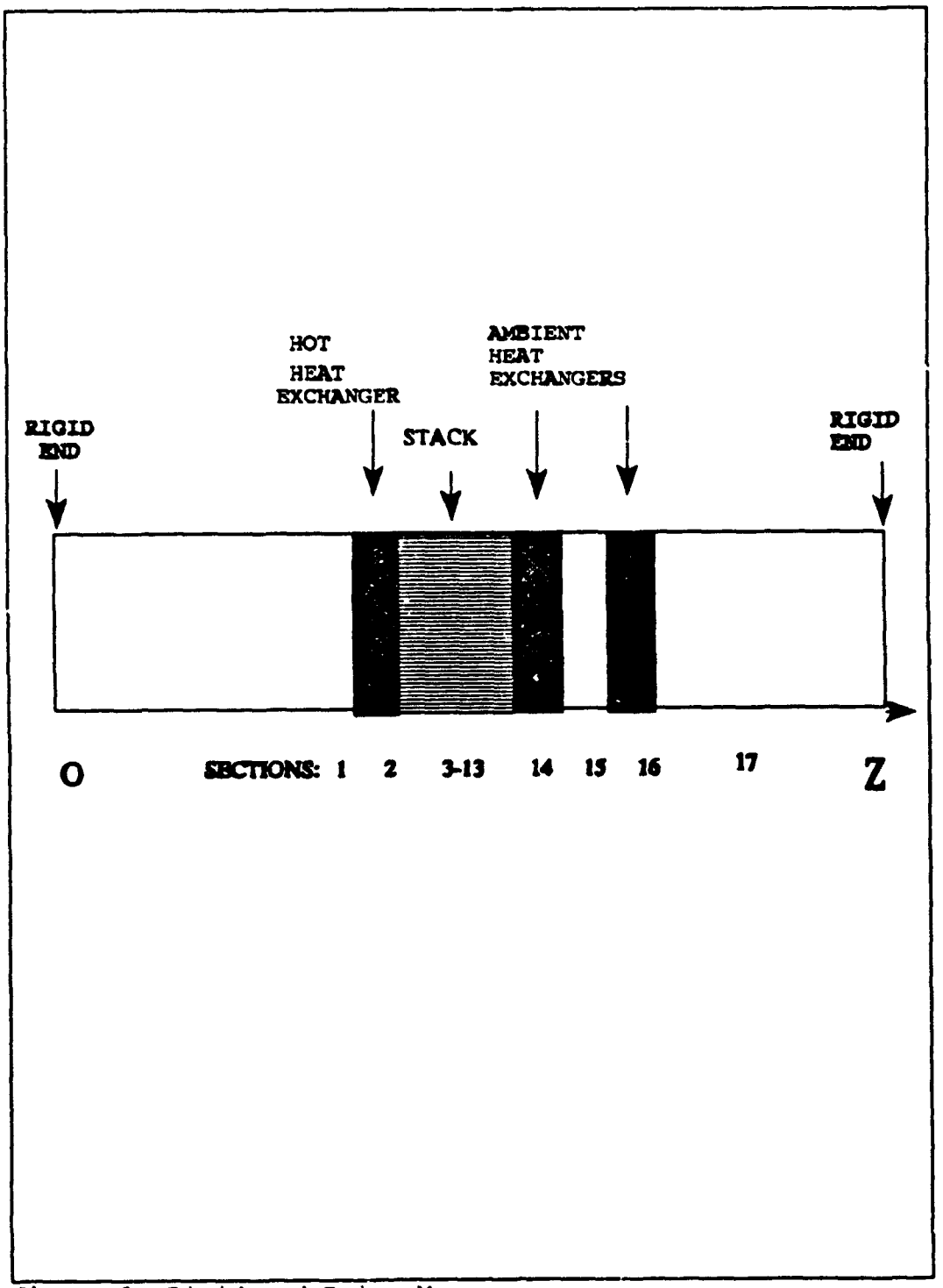

Figure 1. Rigid end Prime Mover

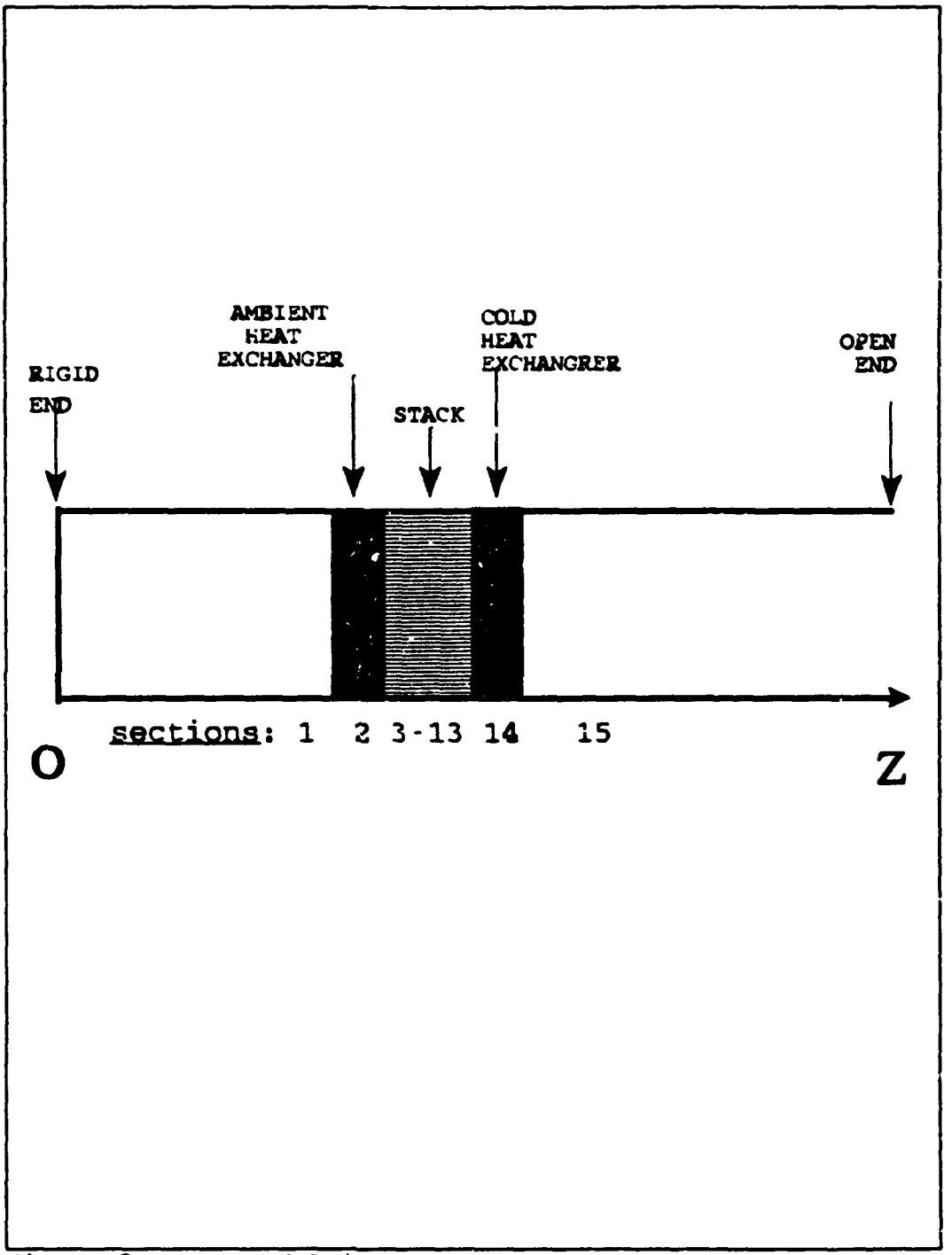

Figure 2. Open end Prime Mover

The computer program finds the normalized acoustic pressure and the specific acoustic impedance in all parts of the prime mover as a function of frequency. The pressure as a function of frequency gives the frequency response. From the frequency response we can determine the quality factor (Q) and the resonance frequency (f_o). To do that we assume a linear temperature gradient across the stack via the hot and ambient (cold) heat exchangers. The stack is divided into eleven subsections. A constant temperature is assumed in each subsection of the stack. In all others parts of the prime mover we assume that the temperature is constant and equal to that of the nearest heat exchanger. The ambient temperature (T_o) is taken to be a function of z ($T_c(z)$) only in the stack area.

The program starts with specific acoustic impedance at the right end. There are two boundary conditions of interest: rigid and open. The specific acoustic impedance of a rigid end is:

$$Z=(1+i)\frac{\sqrt{2}}{2}\frac{1}{(\gamma-1)}\rho_o c\sqrt{\frac{\rho_o c^2}{\omega\eta}}\sqrt{N_{pr}}, \qquad (1)$$

as described in [Ref. 5:p.529]. In the above equation N_{pr} is the Prandtl number, c_p is the isobaric heat capacity per unit mass, γ is the ratio c_p/c_v where c_v is the constant volume heat

capacity per unit mass, η is the viscosity and ρ_o represents the ambient value of the density. The specific acoustic impedance of an open end is:

$$Z = c\rho_0 ka\left(\frac{ka}{4} - 0.6i\right), \tag{2}$$

where a is the radius of the tube and $k=\omega/c$ is the wavenumber, ω is the angular frequency and $i=(-1)^{\frac{1}{2}}$ [Ref. 6:p.202].

The acoustic pressure is specified at the same end. Because the Q is independent of the absolute value of the acoustic pressure in linear theory, the specified pressure is only a relative or normalized pressure. Next we find the pressure and the impedance at the entrance to the hot heat exchanger using the Rayleigh's impedance translation theorem [Ref.1,5]. This is one difference between Arnott's and other methods. That is, other methods find the fluid variables by integrating the governing equations along the prime mover. The impedance translation theorem state that: *within any homogeneous layer, with intrinsic (or characteristic) specific acoustic impedance Z_{int}, the local specific impedance $Z(z-l)$ at $z-l$ is related to that at z by [Ref. 5:p.139]:*

$$Z(z-l) = Z_{int}\frac{Z(y)\cos(kl) - iZ_{int}\sin(kl)}{Z_{int}\cos(kl) - iZ(y)\sin(kl)}. \tag{3}$$

Likewise, the acoustic pressure at the entrance to the hot heat exchanger is found through the pressure translation theorem:

$$P_1(z-l) = P_1(z)\{\cos(kl) - i[\frac{Z_{inc}}{Z(z)}]\sin(kl)\}, \qquad (4)$$

where $Z(z)$ is the specific impedance at z, $P_1(z)$ is the first order value of pressure, i.e, the acoustic pressure, and Z_{inc} is:

$$Z_{inc} = \frac{\rho_0 \omega}{[\Omega F(\lambda) k]}. \qquad (5)$$

In the above equation, Ω represents the porosity and is equal to NA, where A is the cross sectional area of single pore and N is the number of pores per unit area. k is the complex wave number in the pore and given by:

$$(k(\lambda, \lambda_T))^2 = \frac{\omega^2}{c^2} \frac{1}{F(\lambda)} (\gamma - (\gamma - 1) F(\lambda_T)). \qquad (6)$$

The $F(\lambda)$ and $F(\lambda_T)$ factors arise from the equation for the z component of the acoustic velocity:

$$u_z(x, y, z) = \frac{F(x, y; \lambda)}{i\omega \rho_0} \frac{dP_1(z)}{dz}. \qquad (7)$$

The $F(x,y;\lambda)$ satisfies the following differential equation [Ref. 1]:

$$F(x,y;\lambda) + \left(\frac{R^2}{i\lambda^2}\right)\nabla_T^2 F(x,y;\lambda) = 1, \tag{8}$$

where the Laplacian operator ∇^2_T is given by:

$$\nabla_T^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{9}$$

In our case where the pores of the stack are parallel plates, the F depends only on y and the shear wave number λ or the dimensionless thermal disturbance number λ_T which are given by:

$$\lambda = R\left(\frac{\rho_0 \omega}{\eta}\right)^{\frac{1}{2}}, \tag{10}$$

and

$$\lambda_T = \lambda N_{pr}^{\frac{1}{2}}. \tag{11}$$

The $F(y;\lambda)$ is given by:

$$F(y,\lambda) = 1 - \frac{\cosh\left(\sqrt{-1}\frac{\lambda}{2}\frac{y}{a}\right)}{\cosh\left(\sqrt{-1}\frac{\lambda}{2}\right)}. \tag{12}$$

In the above equation a is the half separation distance between the parallel plates of the stack or heat exchanger. The pore average of $F(y;\lambda)$ for heat exchangers and stack, is given by:

$$F(\lambda) = 1 - \left(\frac{2}{\lambda}\sqrt{-i}\right)\tanh\left(\sqrt{-i}\frac{\lambda}{2}\right). \qquad (13)$$

Finally we use the boundary layer approximation in the circular regions of the prime mover:

$$\lim_{\lambda \to \infty} F(\lambda) = 1 - (\delta / R)(1+i), \qquad (14)$$

In the above relation R is twice the hydraulic radius of the pore, or twice the ratio of the transverse pore area to the pore perimeter. Actually for heat exchangers and stack with parallel plates, R is the separation distance (2a) between the plates and for the circular parts it is the radius of the tube. δ is the viscous or thermal penetration depth.

Once P_1 and Z are known at the entrance to heat exchanger, we need the values just inside the heat exchanger. Using the pressure and the impedance translation theorem we have a solution up to the stack where a linear temperature gradients exists. In the heat exchanger three things are different from the circular parts. These are: the wavenumbers in the heat

exchangers are complex, the porosity now is NA, and $F(\lambda)$ as we mention above is given by equation (13).

The point now is how to find pressure and impedance in the stack. In this case Arnott's method uses the following system of the first order differential equations that can be solved using Range-Kutta methods:

$$\frac{dZ(z)}{dz} = ik(z)\,Z_{int}(z)\left(1-\frac{Z(z)^2}{Z_{int}(z)^2}\right)+2\alpha(z)\,Z(z), \qquad (15)$$

and

$$\frac{dP_1(z)}{dz} = ik(z)\,Z_{int}(z)\,\frac{P_1(z)}{Z(z)}, \qquad (16)$$

where $\alpha(z)$ is:

$$\alpha(\lambda, \lambda_T) = \frac{\beta\, T_{0z}}{2}\left[\frac{\frac{F(\lambda_T)}{(F(\lambda))}-1}{1-N_{pr}}\right]. \qquad (17)$$

T_{oz} is the temperature gradient given by:

$$T_{0z} = \frac{dT_0(z)}{dz}. \qquad (18)$$

β is the thermal expansion coefficient:

$$\beta = \frac{-\left(\frac{\partial \rho}{\partial T}\right)_p}{\rho_0}. \qquad (19)$$

In our program, this system of equations is solved with the Runge-Kutta-Fehlberg method for every subsection of the stack. In so doing, the impedance and the pressure distribution in the stack is obtained. Now using the pressure and the impedance translation theorem we find the pressure and the impedance in the rest of the prime mover.

Knowing the pressure at the left end we can calculate the quality factor, the resonance frequency and the maximum pressure amplitude. This is done by minimizing the following equation [Ref. 8]:

$$\frac{\left(\frac{|P(1)|f_0}{Qf}\right)^2}{\left(\frac{f_0}{f}-\frac{f}{f_0}\right)^2+\frac{1}{Q^2}} - amplit^2, \qquad (20)$$

where Q is the quality factor, $|P(1)|$ is the maximum amplitude of pressure at the left end, f_0 is the resonance frequency, amplit is the calculated amplitude of pressure at the left end and is a function of the frequency f. In the above relation there are two known variables f and amplit and three unknowns ($Q, f_o, P(1)$) that we want to find. To find these

unknowns, the 3-parameter least squares technique is used with the help of simplex method.

Up to this point, Q and f_o have been determined only by considering the pressure amplitude at the left end. We have not worried about matching the acoustic impedance at the left end. Impedance matching allows us to refine the values of Q and f_o through the complex eigenfrequency. Using this technique, one assumes the angular frequency is complex and given by:

$$\omega = 2\pi f_o\left(1-\frac{i}{2Q}\right). \qquad (21)$$

Substituting the initial value of f_o and Q determined from the frequency response, into this equation gives an initial guess at the complex eigenfrequency for the system. This frequency is used to calculate the impedance throughout the prime mover as explained earlier. The value of impedance in the gas at the left end is compared to the value for the boundary condition (i.e. the impedance of a rigid end with thermal loss). The difference between the computed and actual impedance is used to adjust the eigenfrequency. The impedance is computed with the new eigenfrequency until both real and imaginary parts of the impedance matche at the left end. The final values of f_o

and Q are determined from the real and imaginary parts of the eigenfrequency that best matche the impedance.

B. IMPLEMENTATION OF ARNOTT'S METHOD

The program uses functions that already exist in Matlab where possible. For the Runge-Kutta-Fehlberg 5th order method we modified the existing function in Matlab to integrate with initial values of the integration variable greater than the final. This is necessary because we put the origin at the left end (z=0) and we begin our calculations from the right end (z=L). One reason that we use this particular Runge-Kutta method is its accuracy. The Runge-Kutta-Fehlberg computes two Runge-Kutta estimates for the value y_{n+1} but of different orders of error. The global error in this numerical method is $O(h^5)$ and the local error is $O(h^6)$ [Ref. 7].

To do the minimization, we use the function fmins with the help of simplex numerical method. The fmin function with simplex method is used to match the impedances at the left end using the technique of least squares.

III. THE MATLAB PROGRAM

The goal of this chapter is to provide an overview of the structure of the Matlab program that implements Arnott's method. Furthermore it describes all the functions, special tricks and limitations of this code. It also explains how to run the program.

A. OUTLINE OF THE MATLAB PROGRAM

The program is divided into three parts. In the first part the variables are specified for every section of the prime mover. They are: the type of section (open, heat exchanger, stack), the number of subsections that exists inside the section, the temperature at the left and right ends of the section, twice the hydraulic radius of the section, and the porosity of the section. Also, specified are the frequency range and the number of intervals in this range.

The second part is the main program. It finds the impedance and the pressure distribution in every section of the prime mover for each frequency within the assumed range. To accomplish this it begins at the right end of the tube where it calculates the impedance of the end, rigid or open, and assigns an arbitrary value to the pressure. Continuing, it calculates using the translation theorems, the pressure and the impedance in all other sections except in the stack. In

the stack, where there exists a temperature gradient, the program calls the function "trode45" (a modified version of the function of Matlab "ode45") to integrate from the right end to the left end of the stack. This function solves differential equations using the 5th order Runge-Kutta numerical method with Fehlberg coefficients.

The last part of the Matlab program uses the calculated pressure at the left end from the previous part to find the maximum amplitude, the resonance frequency and the quality factor. These parameters are found from a least square fit to equation (20). The minimization is performed with the matlab function "fmins" with "options 1,2,3,14". Finally, the Q and f_o estimates are used as a starting points for the final calculation of the quality factor and the resonance frequency using the complex eigenfrequency method. This refinement usually is a small correction (in the fourth decimal place) to initial guesses. Finally, the program plots 1/Q and f_o versus temperature difference and saves the results in a data file.

B. LIMITATIONS OF THE PROGRAM, SPECIAL POINTS.

There are two programs, one for the open end prime mover and one for the rigid end tube. To run the open end progam, one needs to run "arff1.m" with the functions "error,m-erro3.m-argo3f.m". To run the rigid end program one needs to run "argo2.m" with the functions "error.m-erro3.m-argo3.m".

The difference in the two progams is that one uses "argo3.m" and the other uses "argof3.m". Before running one program or the other, one needs to switch these functions in the "erro3.m" function.

Futhermore, there are some factors that influence the operation of the program. One important step is the choice of the initial range of the frequencies. To begin the program one needs to know the approximate resonance frequency as well as which longitudinal acoustic mode is to be used. Initially the frequency range is $f_o \pm 20Hz$. After the program runs for the first value of ΔT the frequency range is automatically adjusted to be:

$$\Delta f = \frac{f_o}{|Q|}. \qquad (22)$$

Another important factor is of course the number of points between the maximum and minimum frequency that the program uses. Usually this number is 500 but sometimes when the values of quality factor gets large this may have to be doubled (1000 points). In these cases, it may also be necessary to use steps of 5 degrees Kelvin instead of the usual 10 degrees Kelvin.

Finally, another important point concerns the "counters" that are used by the program. The most important counter is the "flag". If somebody wants to run the program for a case where the quality factor increases with ΔT, one needs to put the initial value of the "flag" counter equal to zero.

Otherwise, it should be set equal to one. The purpose of this counter is to force the program to scan for negative values of the inverse quality factor at the proper time.

IV. VALIDATION AND RESULTS

In this chapter we compare the results of the program to experimental results for two different prime movers [Ref. 2 and 3] as well as with the results of the standing wave analysis by Atchley [Ref 4].

The first results are for a rigid end prime mover described in Ref. 2 and 4. Comparison of both 1/Q and resonance frequency are made. Figures 3 through 5 show 1/Q vs ΔT (in degrees K) for three different mean gas pressures. The gas is helium. In these figures, the open circles represent measured values, the *'s represent the results of the standing wave analysis and the solid line represents the result of the Matlab program. The overall agreement between the present analysis and the measurements is good, but not as good as for the standing wave analysis. The present analysis agrees best at low values of ΔT. The reasons for the increased discrepancy at higher values of ΔT are not understood. One would expect the present analysis to agree better with measurements than the standing wave analysis. This is expected because the present analysis makes fewer assumptions. In the particular, it does a better job of taking into account changes in the acoustic pressure amplitude in differents parts of the prime mover. The standing wave analysis ignores any changes in

pressure amplitude due to changes in cross sectional area of the prime mover.

Figures 6 through 8 show results of resonance frequency (in Hz) vs ΔT for the same three mean pressures. The agreement between the present analysis and the measured frequencies is within 1% in all cases. The standing wave analysis also agrees with the measured values to about the same accuracy. However, the two methods show different depedences on ΔT. This is because 1) the standing wave analysis considers only the temperature dependence of the sound speed, whereas the present analysis includes the effects of dispersion due to the boundary layers, and 2) the standing wave analysis ignores the finite impedance of the rigid ends. Of these two differences the first is the more important. This results in different curvatures of the two theoretical lines.

The second set of comparisons is made for an open ended prime mover used in Che's thesis research [Ref. 3]. The unusual property of this prime mover is that while 1/Q decreases with ΔT for the first and third modes (as is usual for a prime mover), 1/Q increases with ΔT for the second mode. This behavior provides a good test for theoretical models. The results for 1/Q vs ΔT are shown in figures 9 through 11 for the first, second and third modes, respectively. It is seen that the present analysis and the standing wave analysis are in close agreement in the first and second modes. The agreement diminishes slightly at the third mode. The agreement

with the measured values is worse than before. However, this is most likely due to contamination of the helium with air and water vapor as discuseed in Che's thesis. Another interesting aspect of the analysis is the prediction of an extremum in 1/Q for the second and third modes.

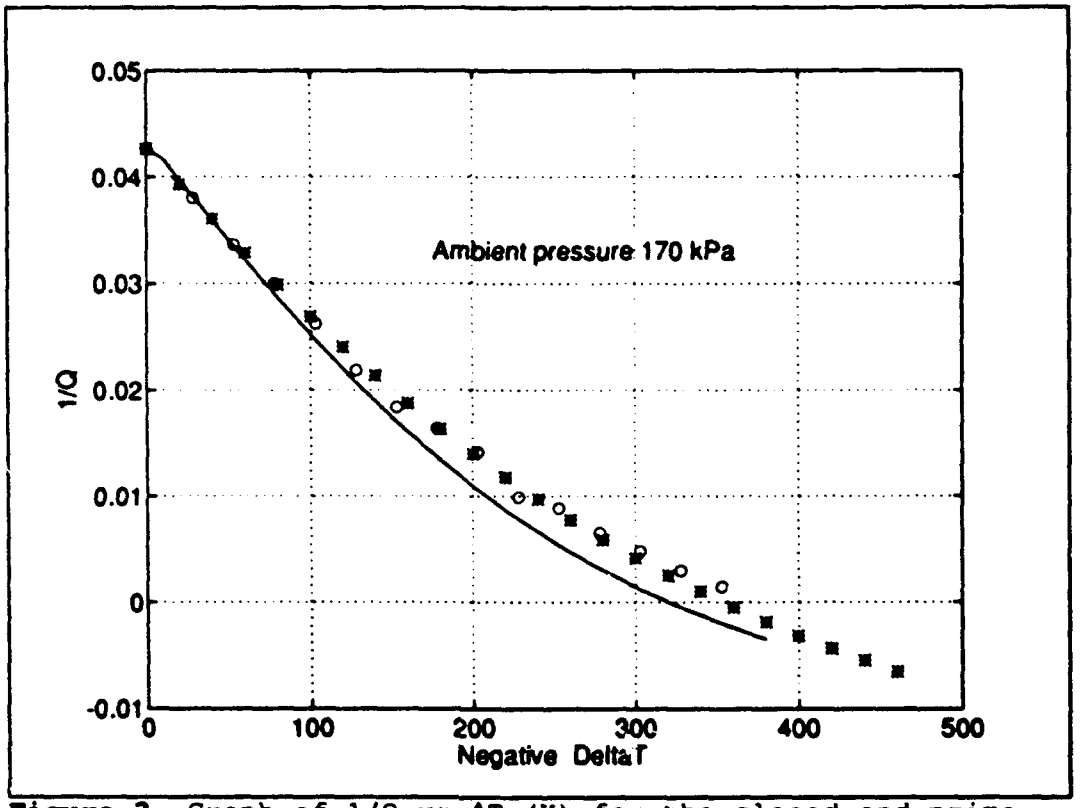

Figure 3. Graph of 1/Q vs ΔT (K) for the closed end prime mover filled with helium at 170 kPa mean pressure.

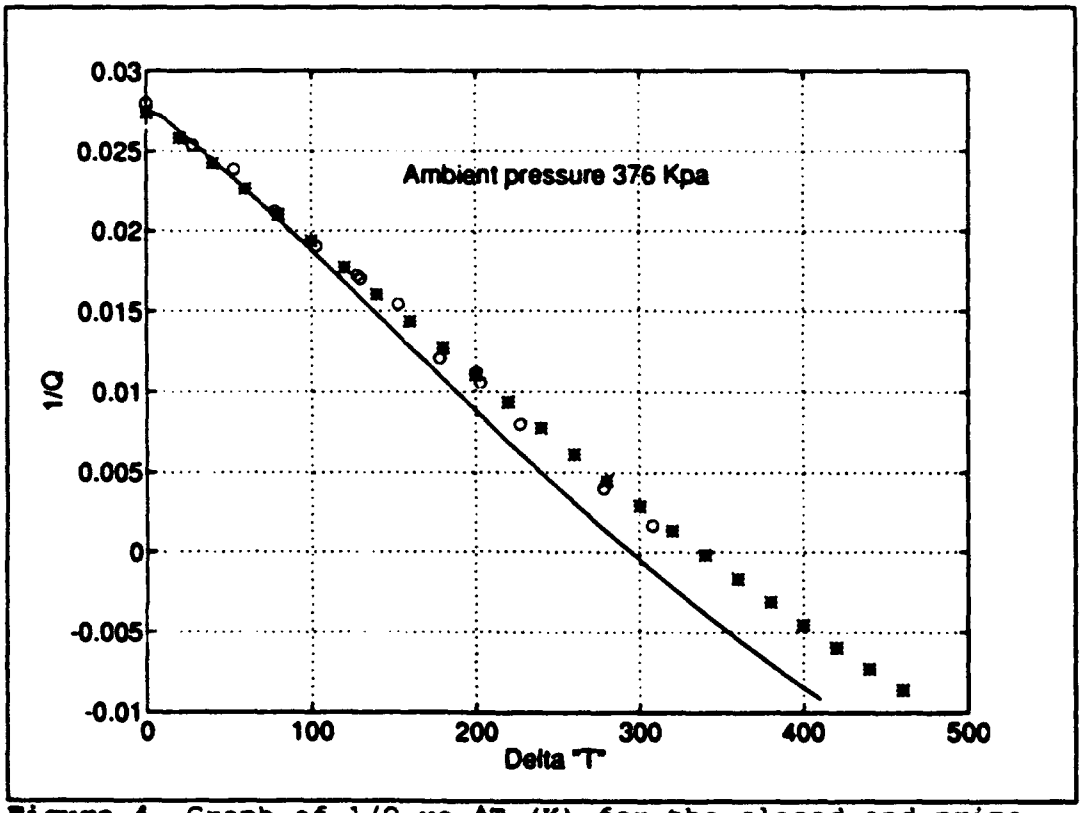

Figure 4. Graph of 1/Q vs ΔT (K) for the closed end prime mover filled with helium at 376 kPa mean pressure.

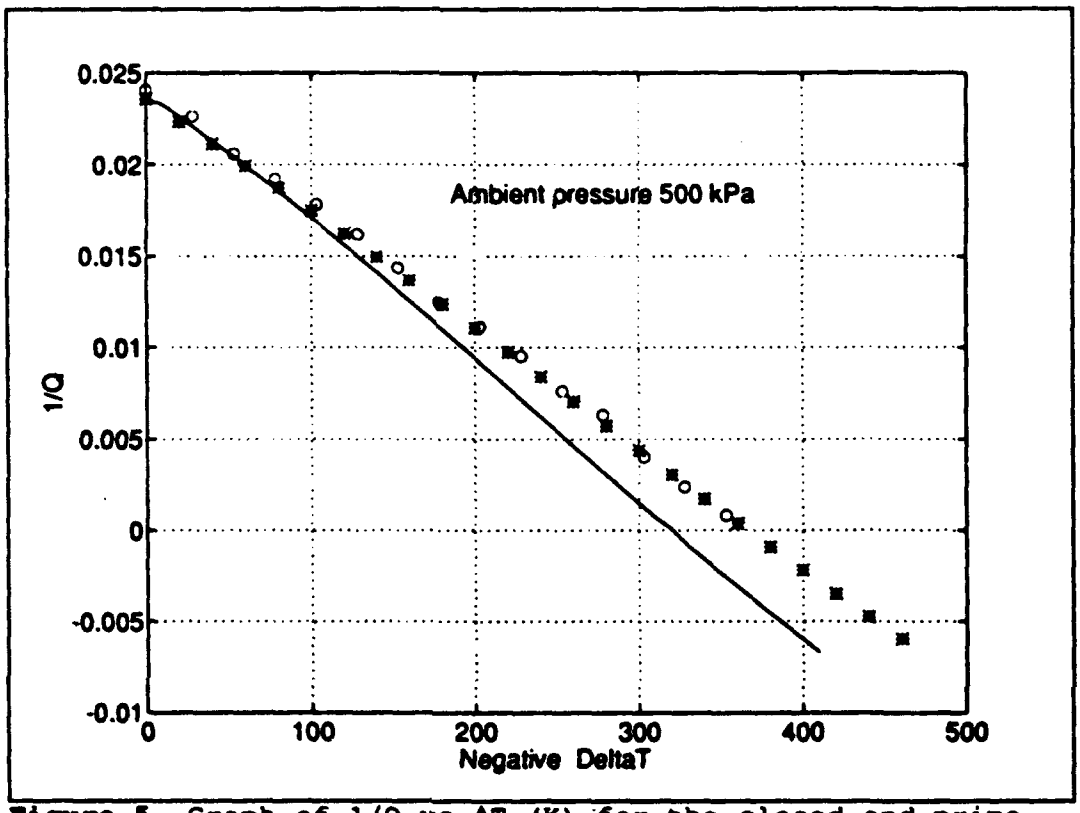

Figure 5. Graph of 1/Q vs ΔT (K) for the closed end prime mover filled with helium at 500 kPa mean pressure.

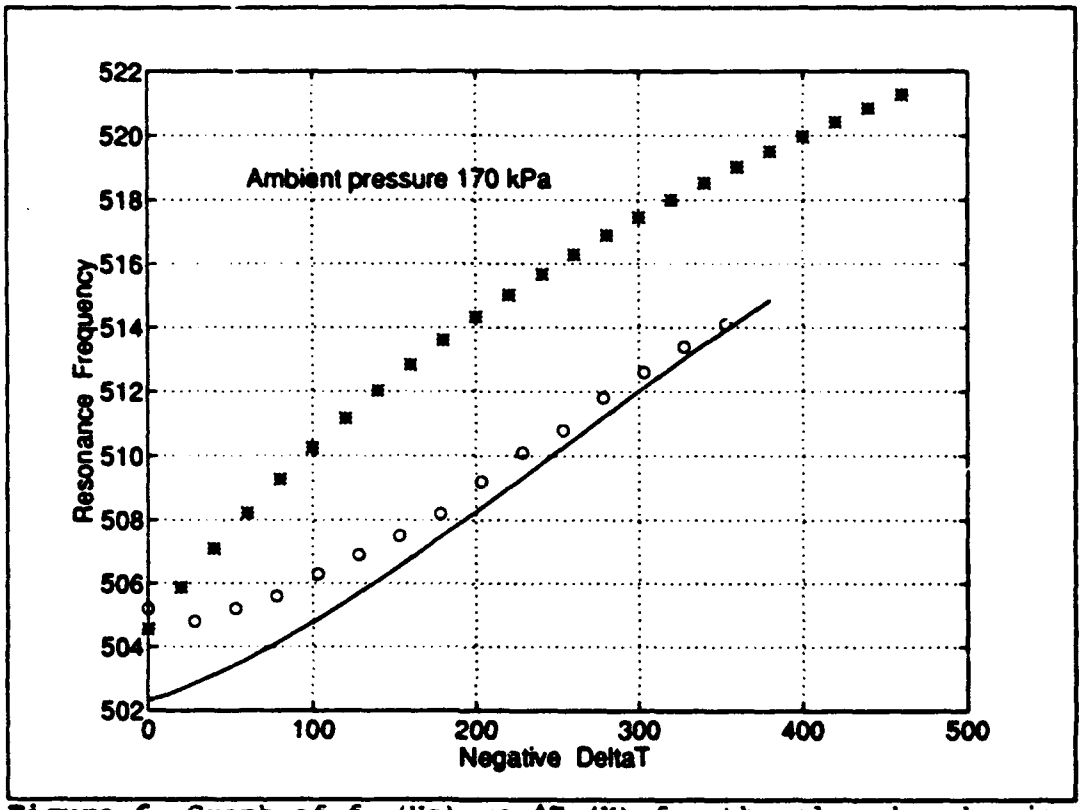

Figure 6. Graph of f_o (Hz) vs ΔT (K) for the closed end prime mover filled with helium at 170 kPa mean pressure.

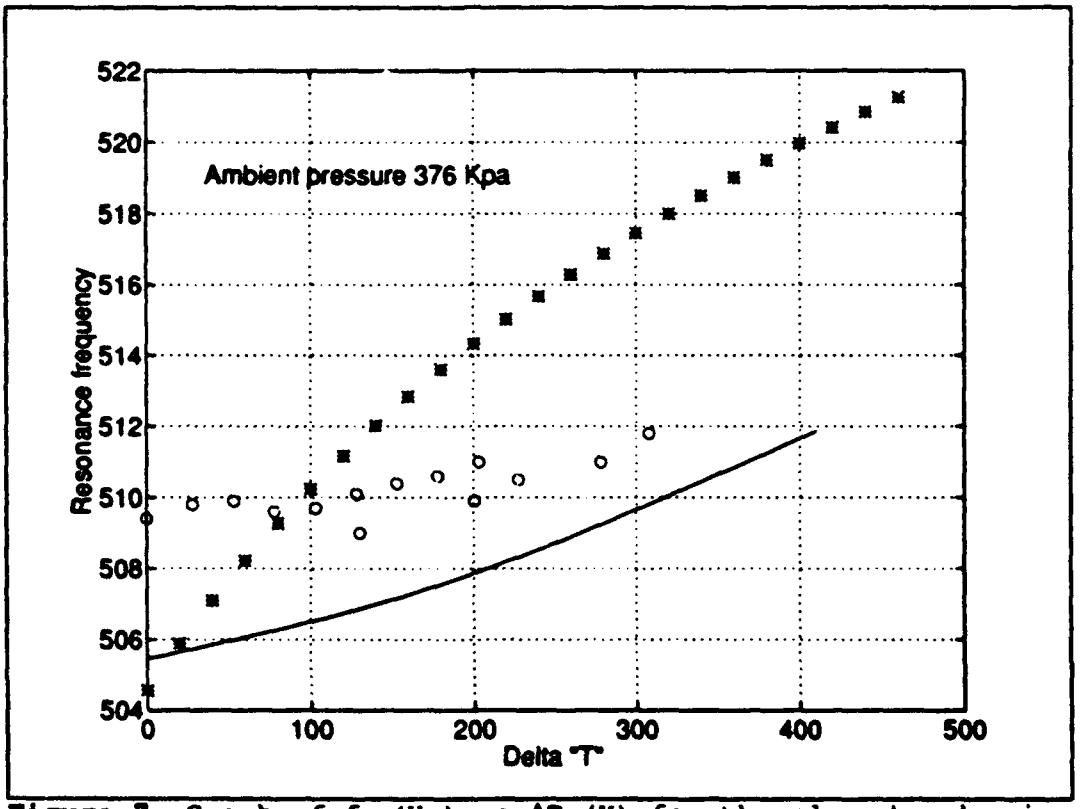

Figure 7. Graph of f_o (Hz) vs ΔT (K) for the closed end prime mover filled with helium at 376 kPa mean pressure.

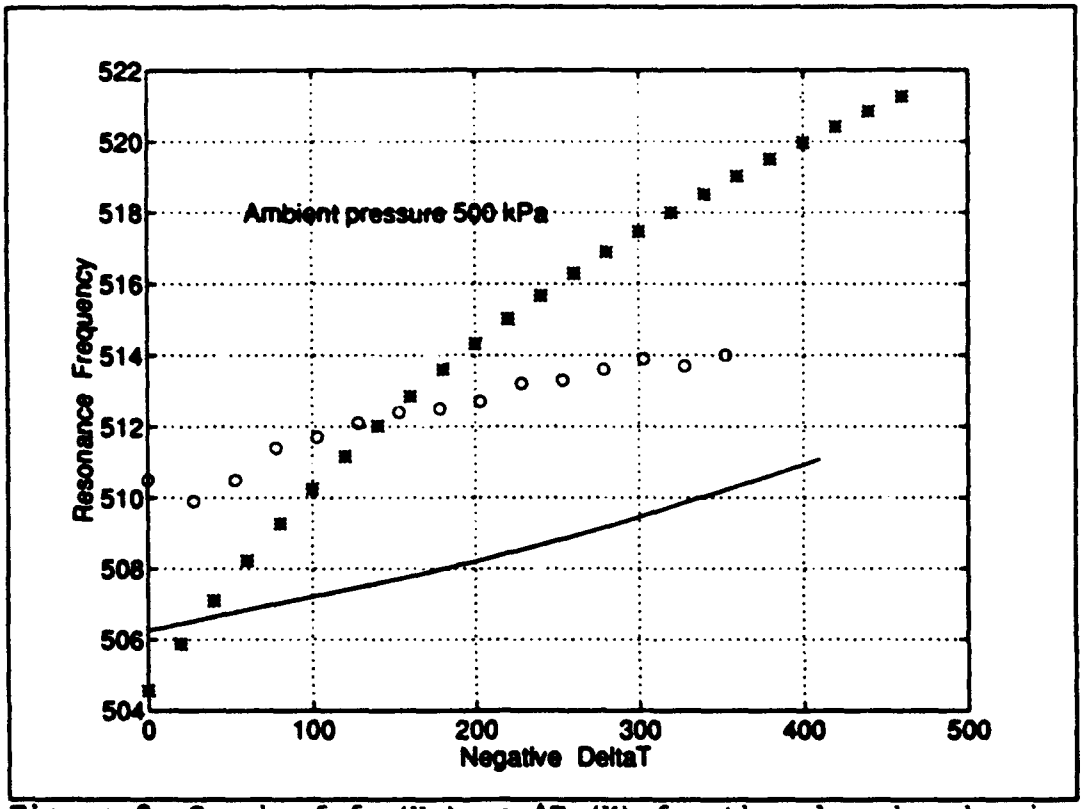

Figure 8. Graph of f_o (Hz) vs ΔT (K) for the closed end prime mover filled with helium at 500 kPa mean pressure.

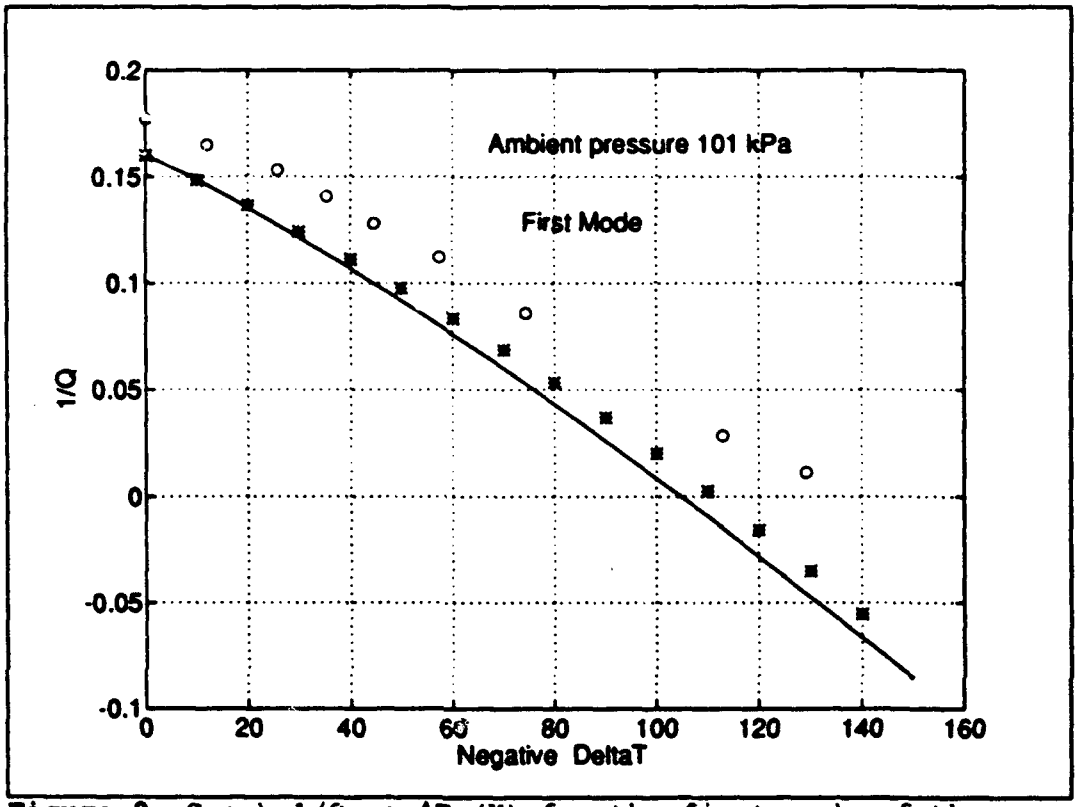

Figure 9. Graph 1/Q vs ΔT (K) for the first mode of the open ended prime mover filled with helium gas at 101 kPa.

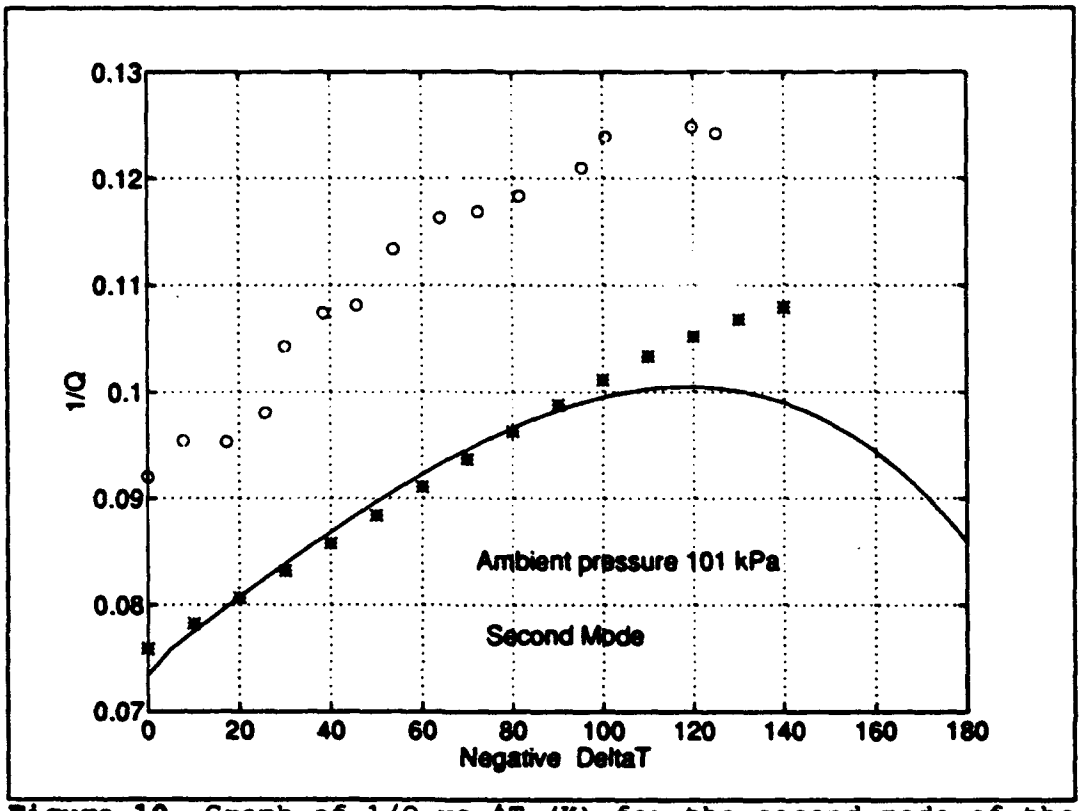

Figure 10. Graph of 1/Q vs ΔT (K) for the second mode of the open ended prime mover filled with helium gas at 101 kPa.

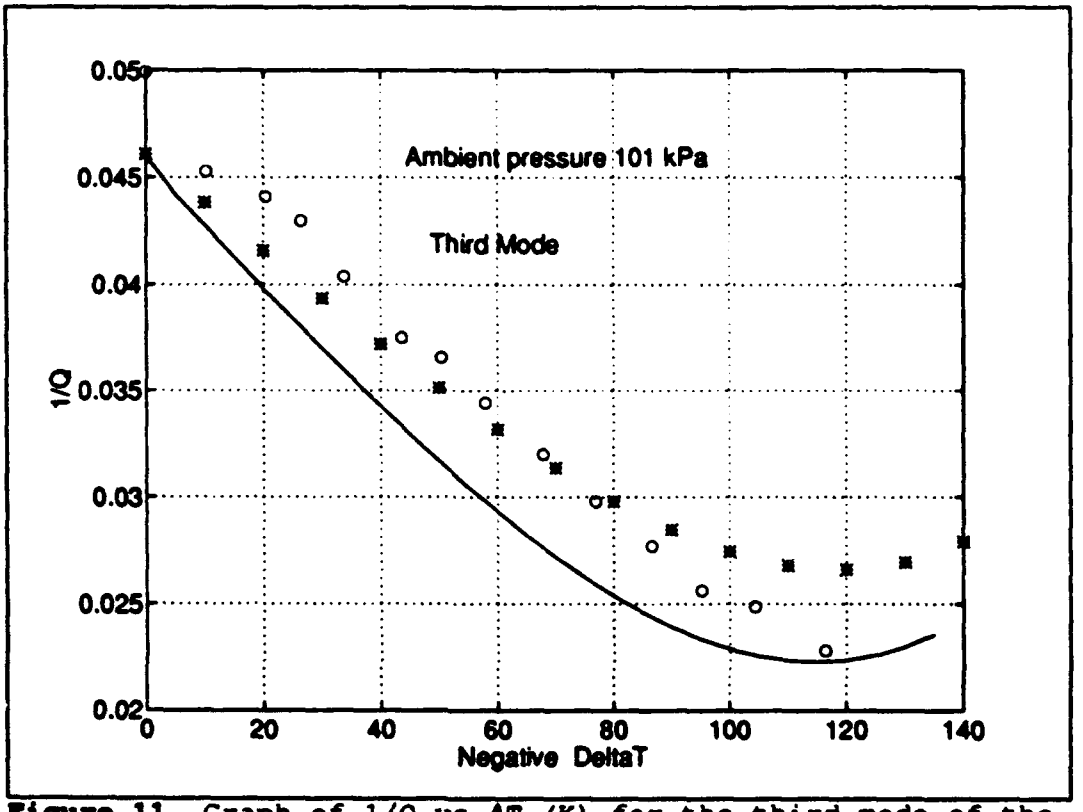

Figure 11. Graph of 1/Q vs ΔT (K) for the third mode of the open ended prime mover filled with helium gas at 101 kPa.

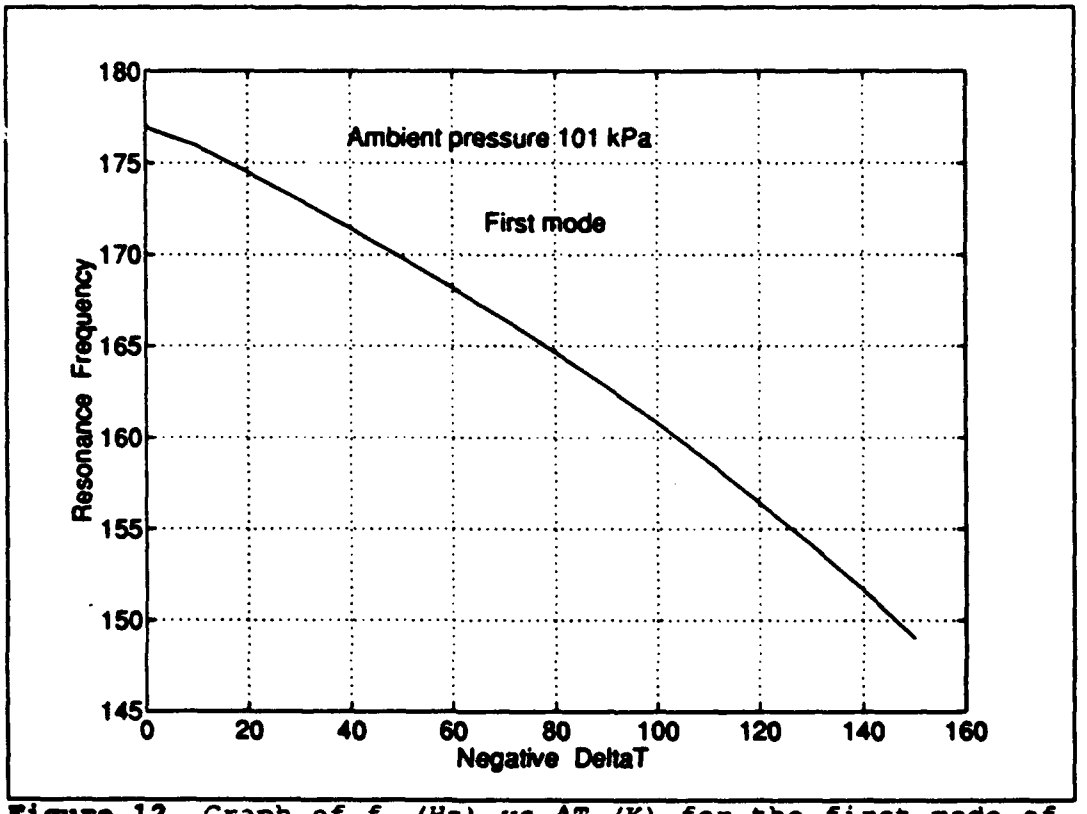

Figure 12. Graph of f_o (Hz) vs ΔT (K) for the first mode of the open ended prime mover filled with helium gas at 101 kPa.

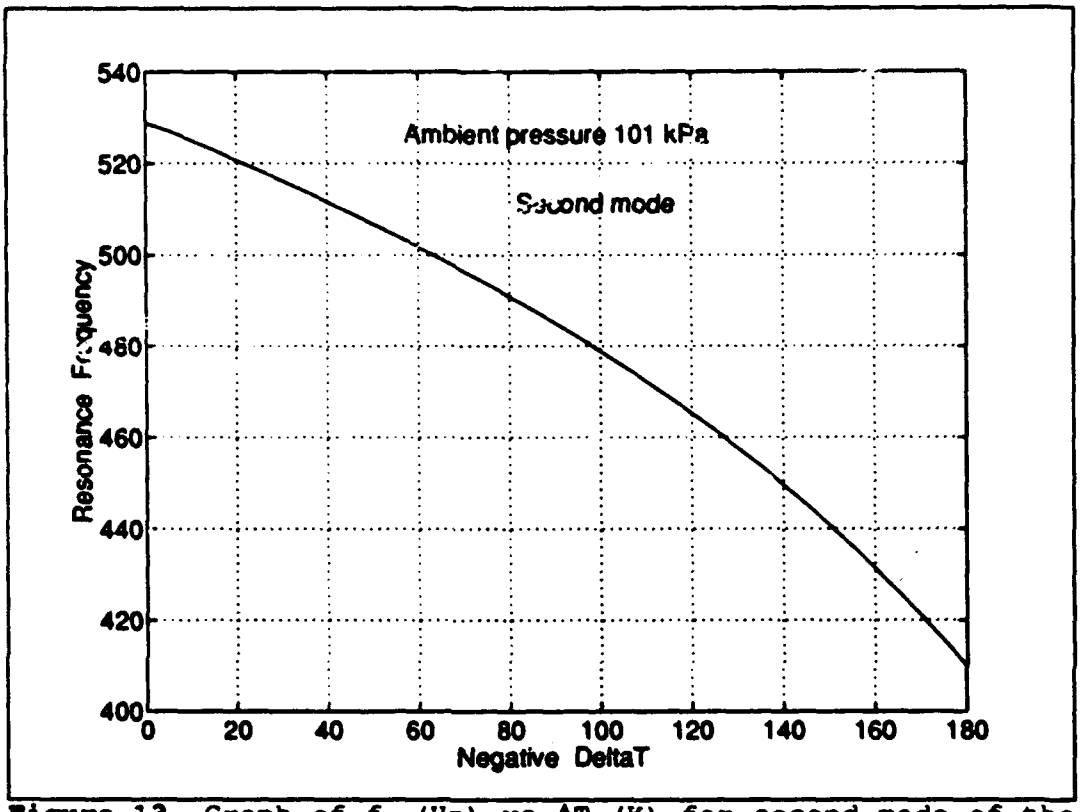

Figure 13. Graph of f_o (Hz) vs ΔT (K) for second mode of the open ended prime mover filled with helium gas at 101 kPa.

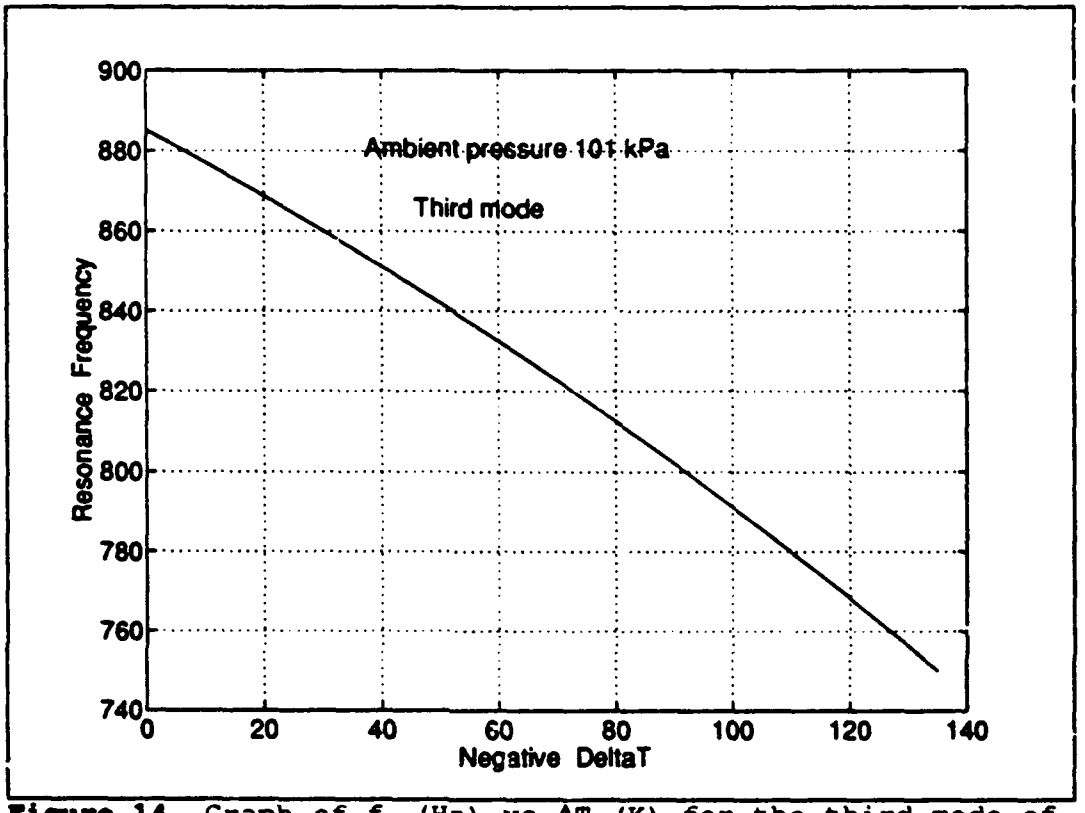

Figure 14. Graph of f_o (Hz) vs ΔT (K) for the third mode of the open ended prime mover filled with helium gas at 101 kPa.

V. SUMMARY

The results of this thesis can be summarized as follows. Arnott's method was implemented in a Matlab progam. The impedance and the pressure distribution were calculated along the prime mover. From the frequency response at the left end and complex eigenfrequency, the quality factor and the resonance frequency was estimated for both cases rigid and open end prime movers. The results of the analysis were compared with those of a standing wave analysis and with experimental results. In the most of the cases the results agree well. A future investigation can be the examination of the output work of the prime mover.

APPENDIX A. THE MATLAB PROGRAM FOR THE OPEN END PRIME MOVER

```
% Program for prime mover.

 % PART 1   (CHANGE)
clear;
clg;
coun=0;                          % counter
pt=0;                            % counter
qual=zeros(45,1);                % quality factor
inqual=zeros(45,1);              % inverse quality factor
fr0=zeros(45,1);                 % resonant frequency
deltaT=zeros(45,1);              % temperature difference
point=0;                         % indicator
flag=1;                          % indicator
slope=sum('negative');
load open.dat                    % standing wave analysis data
% Main Loop for evrey temperature difference (itrgh)
for itrgh=293:-5:158
coun=1+coun
    numfre=1000;                  % number of frequencies
 %  establish some often used constants
    s=5 ;                        % number of sections
j=sqrt(-1);
```

```
npr=2/3*(1+eps);
gamma=5/3*(1+eps);
cp=(2.5*8.3143/(4.e-3))*(1+eps);
ksolid=.16*(1+eps);
sqri=((1+j)/sqrt(2))*(1+eps);
p1=ones(18,numfre).*(10^(-8)); % presure inside the tube
freq=zeros(1,numfre);    % frequncies
w=zeros(1,numfre);       % angular frequencies
teren=zeros(1,s);        % end of sections
numblay=zeros(1,s);      % layers of sections
lengsec=zeros(1,s);      % lengths of sections
tempR=zeros(1,s);        % rigth temperature of sections
tempL=zeros(1,s);        % left temperature of sections
poros=zeros(1,s);        % porocity of stack
ratio=zeros(1,s);        % ratio
% SET VALUES
    fremin=864;              % minimum frequency
    fremax=904;              % maximum frequency
termin=sum(abs('free'));     % end of the last section
ampres=(1.01e+5)*(1+eps);    % ambient presure
    dramp=(1.e-16);          % driver pressure

% section "1"
terenl='opentu';
teren(1,1)=sum(abs(terenl));
numblay(1,1)=1;
```

```
   lengsec(1,1)=(66.25e-2)*(1+eps);
   tempR(1,1)=293*(1+eps);
   tempL(1,1)=293*(1+eps);
   ratio(1,1)=(1.917e-2)*(1+eps);
   poros(1,1)=1*(1+eps);
   a..es(1,:)=pi*ratio(1,:)*(1+eps);
% section "2"
   teren2='hexch';
   teren(1,2)=sum(abs(teren2));
   numblay(1,2)=1;
   lengsec(1,2)=(.82e-2)*(1+eps);
   tempR(1,2)=293*(1+eps);
   tempL(1,2)=293*(1+eps);
   ratio(1,2)=(5.08e-4)*(1+eps);
   poros(1,2)=.666*(1+eps);
 % section "3"
   teren3='stack';
   teren(1,3)=sum(abs(teren3));
   numblay(1,3)=11;
   lengsec(1,3)=(2.59e-2)*(1+eps);
   tempR(1,3)=itrgh*(1+eps);
   tempL(1,3)=293*(1+eps);
   ratio(1,3)=(8.636e-4)*(1+eps);
   poros(1,3)=.89473*(1+eps);
 % section "4"
   teren4='hexch';
```

```
   teren(1,4)=sum(abs(teren4));
   numblay(1,4)=1;
   lengsec(1,4)=(.82e-2)*(1+eps);
   tempR(1,4)=itrgh*(1+eps);
   tempL(1,4)=itrgh*(1+eps);
   ratio(1,4)=(5.08e-4)*(1+eps);
   poros(1,4)=.666*(1+eps);
% section "5"
   teren5='opentu';
   teren(1,5)=sum(abs(teren5));
   numblay(1,5)=1;
   lengsec(1,5)=(.6852)*(1+eps);
   tempR(1,5)=itrgh*(1+eps);
   tempL(1,5)=itrgh*(1+eps);
   ratio(1,5)=(1.917e-2)*(1+eps);
   poros(1,5)=1*(1+eps);

    % PART 2
%  MAIN PROGRAM
% FREQUENCY DISTRIBUTION

   numtot=sum(numblay)+1;
   pt=pt+1;
   if pt~=1
       df=fr0(coun-1)/abs(qual(coun-1));
       fremin=fr0(coun-1)-df;
```

```
        fremax=fr0(coun-1)+df;
    end
    i=1:numfre;
       if i~=1
          freq=fremin;
       else
          freq(1,i)=fremin+(fremax-fremin).*(i./(numfre-1));
       end
        w=2*pi*freq;
  % GET THE SPECIFIC ACOUSTIC IMPENDANCE AND PRESSURE AT ALL
   POINTS.
  % START AT THE RIGHT AN MOVE AT THE LEFT.
dens(1,numtot)=ampres*4.0e-3 /(tempR(1,s)*8.3143); % density
visc(1,numtot)=1.887e-5*(tempR(1,s)/273.15)^(.6567);%
                                                    viscosity
kgas(1,numtot)=visc(1,numtot)*cp/npr; % termal conductivity
sspeed(1,numtot)=972.8*sqrt(tempR(1,s)/273.15); %  speed
                                                 %  isothermal
    type1='free';
    type2='rigid';
        if termin==sum(abs(type2))
   %  IMPEDANCE OF RIGID TERMINATION.
          fac(numtot,1:numfre)=sqrt((dens(1,numtot)...
                *sspeed(1,numtot)^2)./(w.*visc(1,numtot)));
```

```
z(numtot,1:numfre)=(1+j).*(dens(1,numtot).*sspeed(1,numtot).
    ...*fac(numtot,1:numfre)*sqrt(npr))/(sqrt(2)*(gamma-1));
  % IMPEDANCE OF FREE TERMINATION
        else
z(numtot,1:numfre)=free(dens(1,numtot),visc(1,numtot),w,rati
o(1,s)...,sspeed(1,numtot),gamma,npr);
    end
  % IMPEDANCE TRANSLATE FOR THE OPEN TUBE OR HEAT EXCHANGER.
    fault='stack';
    true1='opentu';
    true2='hexch';
      for k=s:-1:1
        jup=0;
        jup=numtot-sum(jup+numblay(1,k:s));
    if teren(1,k)~=sum(abs(fault))
dens(1,jup)=ampres*4.0e-3 /(tempR(1,k).*8.3143); % density
visc(1,jup)=1.887e-5.*(tempR(1,k)/273.15).^(.6567);
                                                % viscosity
kgas(1,jup)=visc(1,jup).*cp/npr;  % termal conductivity
sspeed(1,jup)=972.8.*sqrt(tempR(1,k)/273.15);    %  speed
                                                % isothermal
    %      GET LAMBDA AND LAMBDA(T)
lambda(jup,1:numfre)=ratio(1,k).*sqrt(dens(1,jup).*w./visc(1
,jup));
lambdt(jup,1:numfre)=sqrt(npr).*lambda(jup,1:numfre);
```

```
%      GET F(L), F(L(T)) AND WAVENUMBERS FOR OPEN TUBE
                                                        PARTS
        if teren(1,k)==sum(abs(true1))
 flam(jup,1:numfre)=1-(1+j).*sqrt(2)./lambda(jup,1:numfre);
                                                          % f(l)
 flamt(jup,1:numfre)=1-(1+j).*sqrt(2)./lambdt(jup,1:numfre);
                                                        % f(l(t))
 facl=(1+(gamma-1)/sqrt(npr))/sqrt(2);
 ka(jup,1:numfre)=(w./sspeed(1,jup))...
          .*(1+(1+j).*facl./lambda(jup,1:numfre));
        else
   %      GET F(L), F(L(T)) AND WAVENUMBERS FOR HEAT
EXCHANGERS TYPE "SLIT".
            sqrmi=(1-j)/sqrt(2);
            ar(jup,1:numfre)=sqrmi.*lambda(jup,1:numfre);
            argum(jup,1:numfre)=exp(-ar(jup,1:numfre));
            ar(jup,1:numfre)=ar(jup,1:numfre)/2;
ctanh(jup,1:numfre)=(1-argum(jup,1:numfre))./(1+argum(jup,1:
numfre));
 flam(jup,1:numfre)=1-ctanh(jup,1:numfre)./ar(jup,1:numfre);
                                                        % f(l)
            ar(jup,1:numfre)=sqrmi*lambdt(jup,1:numfre);
            argum(jup,1:numfre)=exp(-ar(jup,1:numfre));
            ar(jup,1:numfre)=ar(jup,1:numfre)/2;
ctanh(jup,1:numfre)=(1-argum(jup,1:numfre))./(1+argum(jup,1:
numfre));
```

```
flamt(jup,1:numfre)=1-ctanh(jup,1:numfre)./ar(jup,1:numfre);
                                                    % f(l(t))
          low=w/sspeed(1,jup);
 ka(jup,1:numfre)=low.*sqrt((gamma-(gamma-1)...
         .*flamt(jup,1:numfre))./flam(jup,1:numfre));
         end
   % TRANSLATION THEOREM
    comden(jup,1:numfre)=dens(1,jup)./flam(jup,1:numfre);
zint(jup,1:numfre)=comden(jup,1:numfre).*w./(ka(jup,1:numfre
).*poros(1,k));
    dsub(1,k)=lengsec(1,k)./numblay(1,k);
    ar(jup,1:numfre)=ka(jup,1:numfre).*dsub(1,k);
    sn(jup,1:numfre)=sin(ar(jup,1:numfre));
    cs(jup,1:numfre)=cos(ar(jup,1:numfre));
    ct(jup,1:numfre)=cs(jup,1:numfre)./sn(jup,1:numfre);
fac3(jup,1:numfre)=zint(jup,1:numfre)./z(jup+1,1:numfre);
    z(jup,1:numfre)=zint(jup,1:numfre).*(ct(jup,1:numfre)...
-j.*fac3(jup,1:numfre))./(fac3(jup,1:numfre).*ct(jup,1:numfr
e)-j);
   p1(jup,1:numfre)=p1(jup+1,1:numfre).*(cs(jup,1:numfre)...
-j.*fac3(jup,1:numfre).*sn(jup,1:numfre));
       else
  % STACK WITH LINEAR DISTRIBUTION OF TEMPERATURE BETWEEN
THE ENDS OF THE STACK
toz=(tempR(1,k)-tempL(1,k))/lengsec(1,k); % approximation
                                                    dz/dt
```

```
    ns=0;
    for la=k+numblay(1,k)-1:-1:k
    ns=ns+1;
    tens=tempR(1,k)-(tempR(1,k)-tempL(1,k))*ns/numblay(1,k);
tnsm1=tempR(1,k)-(tempR(1,k)-tempL(1,k))*(ns-1)/numblay(1,k)
;
tave(1,la)=(tens+tnsm1)/2;
dsub(1,k)=lengsec(1,k)/numblay(1,k);
dens(1,la)=ampres*4.0e-3 /(tave(1,la)*8.3143);     % density
visc(1,la)=1.887e-5*(tave(1,la)/273.15)^(.6567); % viscosity
kgas(1,la)=visc(1,jup)*cp/npr; % termal  conductivity
sspeed(1,la)=972.8*sqrt(tave(1,la)/273.15); %  speed
                                              %  isothermal
    %      GET LAMBDA AND LAMBDA(T)
lambda(la,1:numfre)=ratio(:,k).*sqrt(dens(1,la).*w./visc(1,l
a));
lambdt(la,1:numfre)=sqrt(npr)*lambda(la,1:numfre);
    ar(la,1:numfre)=sqrmi*lambda(la,1:numfre);
    argum(la,1:numfre)=exp(-ar(la,1:numfre));
    ar(la,1:numfre)=ar(la,1:numfre)/2;
ctanh(la,1:numfre)=(1-argum(la,1:numfre))./(1+argum(la,1:num
fre));
flam(la,1:numfre)=1-ctanh(la,1:numfre)./ar(la,1:numfre);
                                                    % f(l)
ar(la,1:numfre)=sqrmi*lambdt(la,1:numfre);
argum(la,1:numfre)=exp(-ar(la,1:numfre));
```

```
ar(la,1:numfre)=ar(la,1:numfre)/2;
ctanh(la,1:numfre)=(1-argum(la,1:numfre))./(1+argum(la,1:num
fre));
flamt(la,1:numfre)=1-ctanh(la,1:numfre)./ar(la,1:numfre);
                                                    % f(l(t))
low=w/sspeed(1,la);
ka(la,1:numfre)=low.*sqrt((gamma-(gamma-1)...
                    .*flamt(la,1:numfre))./flam(la,1:numfre));
zint(la,1:numfre)=dens(1,la).*w...
 ./(poros(1,k).*flam(la,1:numfre).*ka(la,1:numfre));
 alprim(la,1:numfre)=toz*(flamt(la,1:numfre)...
  ./flam(la,1:numfre)-1)./(2*tave(1,la)*(1-npr));
ka1=ka(la,1:numfre).';
zint1=zint(la,1:numfre).';
alpr1=alprim(la,1:numfre).';
zeta0=(lengsec(1,k)-(ns-1)*lengsec(1,k)/numblay(1,k));
zetafin=(lengsec(1,k)-(ns)*lengsec(1,k)/numblay(1,k));
z0=z(la+1,1:numfre).';
tol=1.e-4;
[zeta,zout]=trode45('tube',zeta0,zetafin,z0,tol,ka1,zint1,al
pr1);
n=length(zeta);
z(la,1:numfre)=zout(n,:);
zdumy=z0;
p10=p1(la+1,1:numfre).';
tol=1.e-4;
```

```
[zeta,plout]=trode45('tube1',zeta0,zetafin,p10,tol,ka1,zint1
,zdumy);
n1=length(zeta);
p1(la,1:numfre)=plout(n1,:);
end
end
end

   % MINIMIZATION TO GET RESONANCE FREQUENCY AND QUALITY
FACTOR.
    tfun=-j*w.*z(1,1:numfre)*dramp./p1(1,1:numfre);
    for a=1:s
      p1(a,1:numfre)=p1(a,1:numfre).*tfun;
    end
    amplit=zeros(1,numfre);
    amplit(1,1:numfre)=abs(p1(1,1:numfre));
    [maxamp,i]=max(amplit);
    resfre=freq(i)
    amphaf=maxamp/sqrt(2);
    q=0;
    frehaf=0;
    if point==0
       for s=2:numfre
         if amplit(s)>=amphaf  & amplit(s-1)<=amphaf
            frehaf=(freq(s)+freq(s-1))/2;
            q=0.5*resfre/(resfre-frehaf)
```

```
      end
    end
else
      for s=2:numfre
        if amplit(s)<=amphaf  & amplit(s-1)>=amphaf
          ama=1
           frehaf=(freq(s)+freq(s-1))/2
           q=0.5*resfre/(resfre-frehaf)
            end
      end
  end
if q~=0
   x0=zeros(3,1);
   x0(1,1)=maxamp;
   x0(2,1)=resfre;
   x0(3,1)=q;
   options(1)=0;
   options(2)=1.e-4;
   options(3)=1.e-6;
   options(14)=2000;
   est=fmins('erro',x0,options,[],amplit,freq);
   inqual(coun,1)=est(3,1)^(-1);
   fr0(coun,1)=est(2,1);
   deltaT(coun,1)=abs(itrgh-293);
   qual(coun,1)=est(3,1);
end
```

```
if coun~=1
       dif=inqual(coun-1,1)-inqual(coun,1);
     if dif<0 & coun==2
        slope=sum(abs('positive'));
 elseif dif<0 & coun==3
        slope=sum(abs('positive'));
 end
      if flag==0 &  slope~=sum(abs('positive'))
   if  dif<=0 | q==0
      point=1;
      flag=1;
    for s=2:numfre
          if amplit(s)<=amphaf  & amplit(s-1)>=amphaf
         frehaf=(freq(s)+freq(s-1))/2;
         q=0.5*resfre/(resfre-frehaf)
          end
        end
      x0(1,1)=maxamp;
      x0(2,1)=resfre;
      x0(3,1)=q;
      options(1)=0;
      options(2)=1.e-4;
      options(3)=1.e-6;
      options(14)=2000;
      est=fmins('erro',x0,options,[],amplit,freq);
      inqual(coun,1)=est(3,1)^(-1);
```

```
        fr0(coun,1)=est(2,1);
        deltaT(coun,1)=abs(itrgh-293);
        qual(coun,1)=est(3,1);
         end
        end
      end

        % PART THREE
  % MINIMIZATION USING COMPLEX ANGULAR FREQUENCIES.
      options(1)=0;
      options(2)=1.e-4;
      options(3)=1.e-4;
      options(14)=700;
      w1=2*pi*fr0(coun,1)*(1-j/(2*qual(coun,1)));
   y2=w1+w1*10.e-6;
   y1=w1-w1*10.e-6;
      ella=fmin('erro3',y1,y2,options,itrgh);
   fr0(coun,1)=real(ella)/(2*pi);
      qual(coun,1)=-real(ella)/(2*imag(ella));
   inqual(coun,1)=qual(coun,1)^(-1);
   end
  %     PLOTS AND STORE THE DATA  (CHANGE).
      y=[(inqual(1:coun,1))'; (deltaT(1:coun,1))';
(fr0(1:coun,1))'];
      fid=fopen('inqf3.txt','w+');
fprintf(fid,'Inqual=%-9.6f  T=%-4.1f  RF=%-5.2f\n',y);
```

```
load compf2.dat
    figure
plot(deltaT(1:coun),inqual(1:coun),compf2(:,1),compf2(:,2),'
o'...
            ,open(:,1),open(:,4),'*')
    title('Inverse Quality Factor V.S Temperature
             Difference.Open End');
    grid
    xlabel('Negative Delta "T" ');
    ylabel('1/Q');
    gtext('Ambient pressure 101 Kpa')
        gtext('Experimental Result "o"')
        gtext('Computer Result "*"')
    gtext('Third Mode')
    print -deps inquf3
    figure
    plot(deltaT(1:coun),fr0(1:coun))
    xlabel('Negative Delta "T" ');
    ylabel('Resonance Frequence');
    title('Resonance Frequency V.S Temperature
              Difference.Open End');
    gtext('Ambient pressure 101 Kpa')
    gtext('Third Mode')
    grid
    print -deps frf3
```

```
% FUNCTION TRODE45.M
function ...
[tout,yout]=trode45(FunFcn,t0,tfinal,y0,tol,ka1,zint1,alpr1)
alpha=[1/4 3/8 12/13 1 1/2]';

 beta=[ [    1       0       0     0      0 0]/4
        [    3       9       0     0      0 0]/32
        [ 1932   -7200    7296     0      0 0]/2197
        [ 8341  -32832   29440  -845     0 0]/4104
        [-6080   41040  -28352  9295  -5643 0]/20520]';
 gamma=[ [902880 0 3953664 3855735 -1371249 277020]/7618050
          [ -2090 0   22528   21970   -15048 -27360]/752400
]';
   % initialization
    t=t0;
    hmax=(t-tfinal)/5;
    hmin=(t-tfinal)/2000;
        h=(t-tfinal)/100;
    y=y0(:);
    f=y*zeros(1,6);
    tout=t;
    yout=y.';
    tau=tol*max(norm(y,'inf'),1);
   % Main loop
    while(t>tfinal)&(h>=hmin)
     if t-h>tfinal ,h=t-tfinal; end
```

```
    % Compute the slopes
     temp=feval(FunFcn,ka1,zint1,alpr1,t,y);
     f(:,1)=temp(:);
      for   j=1:5
temp=feval(FunFcn,ka1,zint1,alpr1,t-alpha(j)*h,y-h*f*beta(:,
j));
       f(:,j+1)=temp(:);
      end
    % Estimate the error and the acceptaple error
      delta=norm(h*f*gamma(:,2),'inf');
      tau=tol*max(norm(y,'inf'),1);
    % Update the solution only if the error is acceptaple
      if delta<=tau
     t=t-h;
     y=y-h*f*gamma(:,1);
     tout=[tout;t];
     yout=[yout;y.'];
      end
    % Update the step size
      pow=1/5;
      if delta~=0.0
         h=min(hmax,0.8*h*(tau/delta)^pow);
      end
    end
     if (tfinal<t)
       disp( 'singularity likely.')
```

```
    t
    end
% FUNCTION "TUBE.M" GIVES THE DERIVATIVE OF IMPEDANCE
          function dzdz=tube(ka1,zint1,alpr1,zeta,z)
dzdz=(j.*ka1.*zint1.*(1-(z./zint1).^2)+2.*alpr1.*z);

% FUNCTION "TUBE1.M" GIVES THE DERIVATIVE OF PRESSURE
 function dpdz=tube1(ka1,zint1,zdumy,zeta,p1)
   dpdz=j.*p1.*ka1.*zint1./zdumy;

% FUNCTION "ERRO.M" FOR "FMINS". LEAST SQUARS TECHNIQUE.

      function rmserr=erro(x0,amplit,freq)
    mamp=x0(1,1);
    f0=x0(2,1);
    z=x0(3,1);
 foc1=(mamp*f0)./(z*freq);
 err=((foc1.^2)./((f0./freq-freq/f0).^2+1/z^2))-amplit.^2;
rmserr=sum(err.^2);

 % FUNCTION "ERRO3.M" FOR "FMIN". LEAST SQUARS TECHNIQUE
      function fplus=erro3(w1,itrgh)
  [z,dens,sspeed,visc]=argo3f(w1,itrgh);
  gamma=5/3*(1+eps);
  npr=2/3*(1+eps);
  sqri=((1+j)/sqrt(2))*(1+eps);
```

```
fofw=sqri*sqrt(dens^3*sspeed^4*npr...
              ./(w1.*visc))./(gamma-1);
 s1=real(fofw);
 s2=real(z);
 s3=imag(fofw);
 s4=imag(z);
fplus=(s1^2-s2^2)^2+(s3^2-s4^2)^2;


% FUNCTION "ARGO3F.M". "USED BY ERRO3.M"

   function
[z,dens,sspeed,visc]=argo3(w1,itrgh)
   w=w1;
 %  establish some often used constants

        s=5;          % number of sections
   j=sqrt(-1);
   npr=2/3*(1+eps);
   gamma=5/3*(1+eps);
   cp=(2.5*8.3143/(4.e-3))*(1+eps);
   ksolid=.16*(1+eps);
   sqri=((1+j)/sqrt(2))*(1+eps);
   teren=zeros(s,1);
   numblay=zeros(1,s);
   lengsec=zeros(1,s);
```

```
  tempR=zeros(1,s);
  tempL=zeros(1,s);
  poros=zeros(1,s);
  ratio=zeros(1,s);
% SET VALUES
  termin=sum(abs('free'));
  ampres=(1.01e+5)*(1+eps);
      dramp=(1.e-8);


% section "1"
  teren1='opentu';
  teren(1,1)=sum(abs(teren1));
  numblay(1,1)=1;
  lengsec(1,1)=(65.34e-2)*(1+eps);
  tempR(1,1)=293*(1+eps);
  tempL(1,1)=293*(1+eps);
  ratio(1,1)=(1.917e-2)*(1+eps);
  poros(1,1)=1*(1+eps);
  ares(1,:)=pi*ratio(1,:)*(1+eps);
% section "2"
  teren2='hexch';
  teren(1,2)=sum(abs(teren2));
  numblay(1,2)=1;
  lengsec(1,2)=(.82e-2)*(1+eps);
  tempR(1,2)=293*(1+eps);
```

```
tempL(1,2)=293*(1+eps);
ratio(1,2)=(5.08e-4)*(1+eps);
poros(1,2)=.666*(1+eps);
% section "3"
teren3='stack';
teren(1,3)=sum(abs(teren3));
numblay(1,3)=11;
lengsec(1,3)=(2.59e-2)*(1+eps);
tempR(1,3)=itrgh*(1+eps);
tempL(1,3)=293*(1+eps);
ratio(1,3)=(8.636e-4)*(1+eps);
poros(1,3)=.89473*(1+eps);
% section "4"
teren4='hexch';
teren(1,4)=sum(abs(teren4));
numblay(1,4)=1;
lengsec(1,4)=(.82e-2)*(1+eps);
tempR(1,4)=itrgh*(1+eps);
tempL(1,4)=itrgh*(1+eps);
ratio(1,4)=(5.08e-4)*(1+eps);
poros(1,4)=.666*(1+eps);
% section "5"
teren5='opentu';
teren(1,5)=sum(abs(teren5));
numblay(1,5)=1;
lengsec(1,5)=(.6852)*(1+eps);
```

```
   tempR(1,5)=itrgh*(1+eps);
   tempL(1,5)=itrgh*(1+eps);
   ratio(1,5)=(1.917e-2)*(1+eps);
   poros(1,5)=1*(1+eps);
    if s>=7
% section "6"
   teren6='hexch';
   teren(1,6)=sum(abs(teren6));
   numblay(1,6)=1;
   lengsec(1,6)=(1.02e-2)*(1+eps);
   tempR(1,6)=293*(1+eps);
   tempL(1,6)=293*(1+eps);
   ratio(1,6)=(1.02e-3)*(1+eps);
   poros(1,6)=.667*(1+eps);
% section "7"
   teren7='opentu';
   teren(1,7)=sum(abs(teren7));
   numblay(1,7)=1;
   lengsec(1,7)=(5.483e-2)*(1+eps);
   tempR(1,7)=293*(1+eps);
   tempL(1,7)=293*(1+eps);
   ratio(1,7)=1.917e-2*(1+eps);
   poros(1,7)=1*(1+eps);
     end

  % START AT THE RIGHT AND MOVE AT THE LEFT.
```

```
    numtot=sum(numblay)+1;
dens(1,numtot)=ampres*4.0e-3 /(tempR(1,s)*8.3143); % density
visc(1,numtot)=1.887e-5*(tempR(1,s)/273.15)^(.6567);
                                                  % viscosity
kgas(1,numtot)=visc(1,numtot)*cp/npr; % termal  conductivity
sspeed(1,numtot)=972.8*sqrt(tempR(1,s)/273.15); %  speed
                                                % isothermal
    type1='free';
    type2='rigid';
        if termin==sum(abs(type2))
   %  IMPEDANCE OF RIGID TERMINATION.
 fac3(1,numtot)=sqrt((dens(1,numtot)...
                *sspeed(1,numtot).^2)./(w.*visc(1,numtot)));
z(1,numtot)=(1+j).*(dens(1,numtot).*sspeed(1,numtot)...
           .*fac3(1,numtot)*sqrt(npr))/(sqrt(2)*(gamma-1));
  % IMPEDANCE OF FREE TERMINATION
        else
z(1,numtot)=free(dens(1,numtot),visc(1,numtot),w,ratio(1,s).
..         ,sspeed(1,numtot),gamma,npr);
    end

  % IMPEDANCE TRANSLATE FOR THE OPEN TUBE OR HEAT EXCHANGER.
    fault='stac';
    true1='opentu';
    true2='hexch';
      for k=s:-1:1
```

```
        jup=0;
        jup=numtot-sum(jup+numblay(1,k:s));
    if teren(1,k)~=sum(abs(fault))
dens(1,jup)=ampres*4.0e-3 /(tempR(1,k).*8.3143); % density
visc(1,jup)=1.887e-5.*(tempR(1,k)/273.15).^(.6567);
                                                  % viscosity
kgas(1,jup)=visc(1,jup).*cp/npr;  % termal  conductivity
sspeed(1,jup)=972.8.*sqrt(tempR(1,k)/273.15);    %  speed
                                              %  isothermal


   %      GET LAMBDA AND LAMBDA(T)
lambda(1,jup)=ratio(1,k).*sqrt(dens(1,jup).*w./visc(1,jup));
        lambdt(1,jup)=sqrt(npr).*lambda(1,jup);
   %      GET F(L), F(L(T)) AND WAVENUMBERS FOR OPEN TUBE
PARTS
        if teren(1,k)==sum(abs(true1))
flam(1,jup)=1-(1+j).*sqrt(2)./lambda(1,jup);    %        f(l)
flamt(1,jup)=1-(1+j).*sqrt(2)./lambdt(1,jup);  % f(l(t))
fac31=(1+(gamma-1)/sqrt(npr))/sqrt(2);
ka(1,jup)=(w./sspeed(1,jup))...
                          .*(1+(1+j).*fac31./lambda(1,jup));
        else
   %      GET F(L), F(L(T)) AND WAVENUMBERS FOR HEAT
EXCHANGERS TYPE "SLIT".
           sqrmi=(1-j)/sqrt(2);
           ar(1,jup)=sqrmi.*lambda(1,jup);
```

```
            argum(1,jup)=exp(-ar(1,jup));
            ar(1,jup)=ar(1,jup)/2;
            ctanh(1,jup)=(1-argum(1,jup))./(1+argum(1,jup));
            flam(1,jup)=1-ctanh(1,jup)./ar(1,jup);    % f(l)
            ar(1,jup)=sqrmi*lambdt(1,jup);
            argum(1,jup)=exp(-ar(1,jup));
            ar(1,jup)=ar(1,jup)/2;
            ctanh(1,jup)=(1-argum(1,jup))./(1+argum(1,jup));
            flamt(1,jup)=1-ctanh(1,jup)./ar(1,jup); % f(l(t))
            low=w/sspeed(1,jup);
            ka(1,jup)=low.*sqrt((gamma-(gamma-1)...
                                .*flamt(1,jup))./flam(1,jup));
        end
% TRANSLATION THEOREM
 comden(1,jup)=dens(1,jup)./flam(1,jup);
 zint(1,jup)=comden(1,jup).*w./(ka(1,jup).*poros(1,k));
 dsub(1,k)=lengsec(1,k)./numblay(1,k);
 ar(1,jup)=ka(1,jup).*dsub(1,k);
 sn(1,jup)=sin(ar(1,jup));
 cs(1,jup)=cos(ar(1,jup));
 ct(1,jup)=cs(1,jup)./sn(1,jup);
 fac3(1,jup)=zint(1,jup)./z(1,1+jup);
 z(1,jup)=zint(1,jup).*(ct(1,jup)...
          -j.*fac3(1,jup))./(fac3(1,jup).*ct(1,jup)-j);
    else
```

```
   %  STACK WITH LINEAR DISTRIBUTION BETWEEN THE ENDS OF THE
STACK.
    toz=(tempR(1,k)-tempL(1,k))/lengsec(1,k);    % approx
dz/dt
    ns=0;
    for la=k+numblay(1,k)-1:-1:k
    ns=ns+1;
    tens=tempR(1,k)-(tempR(1,k)-tempL(1,k))*ns/numblay(1,k);
tnsm1=tempR(1,k)-(tempR(1,k)-tempL(1,k))*(ns-1)/numblay(1,k)
;
tave(1,la)=(tens+tnsm1)/2;
dsub(1,k)=lengsec(1,k)/numblay(1,k);
dens(1,la)=ampres*4.0e-3 /(tave(1,la)*8.3143);    % density
    visc(1,la)=1.887e-5*(tave(1,la)/273.15)^(.6567);  %
viscosity
kgas(1,la)=visc(1,jup)*cp/npr; % termal conductivity
sspeed(1,la)=972.8*sqrt(tave(1,la)/273.15);       %  speed
                                                  %  isothermal
     %      GET LAMBDA AND LAMBDA(T)
lambda(1,la)=ratio(1,k).*sqrt(dens(1,la).*w./visc(1,la));
    lambdt(1,la)=sqrt(npr)*lambda(1,la);
    ar(1,la)=sqrmi*lambda(1,la);
    argum(1,la)=exp(-ar(1,la));
    ar(1,la)=ar(1,la)/2;
    ctanh(1,la)=(1-argum(1,la))./(1+argum(1,la));
    flam(1,la)=1-ctanh(1,la)./ar(1,la);          % f(l)
```

```
      ar(1,la)=sqrmi*lambdt(1,la);
      argum(1,la)=exp(-ar(1,la));
      ar(1,la)=ar(1,la)/2;
      ctanh(1,la)=(1-argum(1,la))./(1+argum(1,la));
      flamt(1,la)=1-ctanh(1,la)./ar(1,la);        % f(l(t))
      low=w/sspeed(1,la);
          ka(1,la)=low.*sqrt((gamma-(gamma-1)...
                        .*flamt(1,la))./flam(1,la));
          zint(1,la)=dens(1,la).*w...
                          ./(poros(1,k).*flam(1,la).*ka(1,la));
          alprim(1,la)=toz*(flamt(1,la)...
                  ./flam(1,la)-1)./(2*tave(1,la)*(1-npr));
      ka1=ka(1,la).';
      zint1=zint(1,la).';
      alpr1=alprim(1,la).';
      zeta0=(lengsec(1,k)-(ns-1)*lengsec(1,k)/numblay(1,k));
      zetafin=(lengsec(1,k)-(ns)*lengsec(1,k)/numblay(1,k));
      z0=z(1,1+la).';
      tol=1.e-4;
[zeta,zout]=trode45('tube',zeta0,zetafin,z0,tol,ka1,zint1,al
pr1);
      n=length(zout);
      z(1,la)=zout(n,1).';
      end
        end
       end
```

```
z=z(1,1);
dens=dens(1,1);
sspeed=sspeed(1,1);
visc=visc(1,1);
```

LIST OF REFERENCES

1. Arnott, W.P. , H.E.Bass, and R.Raspet, "General formulation of thermoacoustics for stacks having arbitrarily shaped pore cross sections", *J. Acoustic. Soc. Am., v.90(6), 3228-3237 (1991).*

2. Lin, H.T. , "Investigation of a Heat Driven Thermoacoustic Prime Mover," Master's Thesis, Naval Postgraduate School, Monterey, California, December 1989.

3. Che, C.H. , "Investigation of Thermoacoustic Muffler," Master's Thesis, Naval Postgraduate School, Monterey, California, December 1992.

4. Atchley, A.A. , "Standing wave analysis of a thermoacoustic prime mover below onset of self-oscillation," *J. Acoustic. Soc. Am., v.92(5), 2907-2914 (1992).*

5. Pierce, A.D. , *Acoustics. An Introduction to Its Physical Principles and Applications,* Acoustical Society of America, 1989.

6. Kinsler, L.E. , A.R.Frey, A.B.Coppens, J.V.Sanders, *Fundamentals of Acoustics,* John Wiley & Sons, New York, 1982.

7. Curtis, F.G. , Patrick, O.W. , *Applied Numerical Analysis,* 4th ed., Addison Wesley, 1992.

8. French, A.P. , *Vibration and Waves,* Norton, 1971

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria VA 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey CA 93943-5002	2
3.	Prof. Anthony A. Atcheley, code PH/Ay Department of Physics Naval Postgraduate School Monterey, California 93943	6
4.	Dr. Felipe Gaitan Department of Physics Naval Postgraduate School Monterey, California 93943	1
5.	LTJG. Argirios Gamaletsos 26 Klious, 155-61 Holargos Athens, Greece	2
6.	Prof. W.P. Colson, Chairman, Physics Department Naval Postgraduate School Monterey, California 93943	1